Compulsory 1 - STAT250

Sigbjørn Fjelland

18/3/2022

```
library(tinytex)
library(ggplot2)
```

```
## Warning in register(): Can't find generic `scale_type` in package ggplot2 to
## register S3 method.
```

```
library(reshape2)
```

Task-1

We have that:

such that:

$$r = 1 \rightarrow r^2 = 1 \tag{1}$$

$$A_{quarter} = \frac{\pi \cdot r^2}{4} \Rightarrow \frac{\pi}{4} \tag{2}$$

$$A_{\text{square box}} = r^2 = 1 \tag{3}$$

We want to use the raindrop method using the proportion of hits within the circle relative to the proportion hitting the box in total to estimate π

$$E[\pi] = P(\text{"hit the circle"}) \cdot X \tag{4}$$

$$P(\text{"hit the circle"}) = \frac{A_{quarter}}{A_{\text{square box}}} \tag{5}$$

combine (2), (3), and (4) and we have that:

$$\begin{aligned} P(\text{"hit the circle"}) &= \frac{\frac{\pi \cdot r^2}{4}}{r^2} \\ &= \frac{\pi \cdot r^2}{4 \cdot r^2} \\ &= \frac{\pi}{4} \end{aligned}$$

hence

$$\Rightarrow \pi = 4 \cdot P(\text{"hit the circle"})$$

which is formula (4) with X=4

we need a criteria to sort out which hits within the circle and which misses. To do so we simulate the position of each drop (x, y) from the uniform distribution and apply it to pytagoras:

$$\begin{aligned} x^2 + y^2 &= r^2 \\ x^2 + y^2 &\leq r^2 \\ \Rightarrow \sqrt{x^2 + y^2} &\leq r \end{aligned}$$

such that everything that falls within r maps to the cirkle.

from (3) we have that $r^2 = r = 1$

the relative proportion that falls within r to the total of simulations falling within the total square (5) give the probability of hitting the quarter square

```
estimating_pi <- function(n){
  #simulating position
  x <- runif(n)
  y <- runif(n)

  r = sqrt(x^2+y^2) #hitting criteria

  P_hit_miss <- length(which(r <= 1))/n #relative proportion hit/miss

  pi_emp <- 4*P_hit_miss

  return(pi_emp)
}
estimating_pi(100000)
```

```
## [1] 3.14592
```

```
set.seed(12345)
n=1000 ; m=10000
estimate_of_estimates_of_pi <- c()

for(i in 1:n){
  estimate_of_estimates_of_pi[i] <- estimating_pi(m)

  }
mean_of_estiamts <- mean(estimate_of_estimates_of_pi)
sd_of_estimats <- sd(estimate_of_estimates_of_pi)

cat("as we se the mean of", n, "x", m, "simulations is ", mean_of_estiamts,
    " which is ", "\n", (mean_of_estiamts-pi),
    " from true pi, and a standard deviation of ",sd_of_estimats)
```

```
## as we se the mean of 1000 x 10000 simulations is  3.141359  which is
##  -0.0002334536  from true pi, and a standard deviation of  0.01610762
```

This obviously converge towards true π

Task-2

Sample from uniform distribution and Group the cdf according to spinner:

$$\begin{aligned} U =& \{u_1, u_2, ..., u_n\} \\ U \sim& U(0,1) \\ Y =& U \cdot 1_{\{0 \leq u \leq 0.5\}} \\ R =& U \cdot 1_{\{0.5 < u \leq 0.75\}} \\ B =& U \cdot 1_{\{0 < u \leq 1\}} \end{aligned}$$

multiply a vector of the number of spinn hitts acc to group (color) with a vector of points:

$$\begin{aligned} X &= \{1, -1, 2\} \\ Z &= \{n(Y), n(R), n(B)\} \\ \text{total points} &= Z \cdot X \end{aligned}$$

```
set.seed(1234)

spinn <- function(n){
    P = c(0.5, 0.75, 1); #cdf groupt acc. to spinner
    X = c(1, -1, 2);
    U <- runif(n);

    Y <- which(U<P[1])
    R <- which(P[1]<= U & U< P[2])
    B <- which(P[2]< U & U <=P[3])

    Z <- c(length(Y),length(R),length(B))
    point <- sum(Z*X)
    return(point)
}

#iterate n spinns m times:
P_negative <- function(n, m){
  negative_points <- c()
  for(i in 1:m){
        negative_points <-c(negative_points, spinn(n))
        }

  return(mean(negative_points<0))
}

n = 10;  m=1000;   score = P_negative(n,m)

cat('Over', m, 'trails of',n,'spinns the probabilitie
    of scoring a negative sum is',score)
```

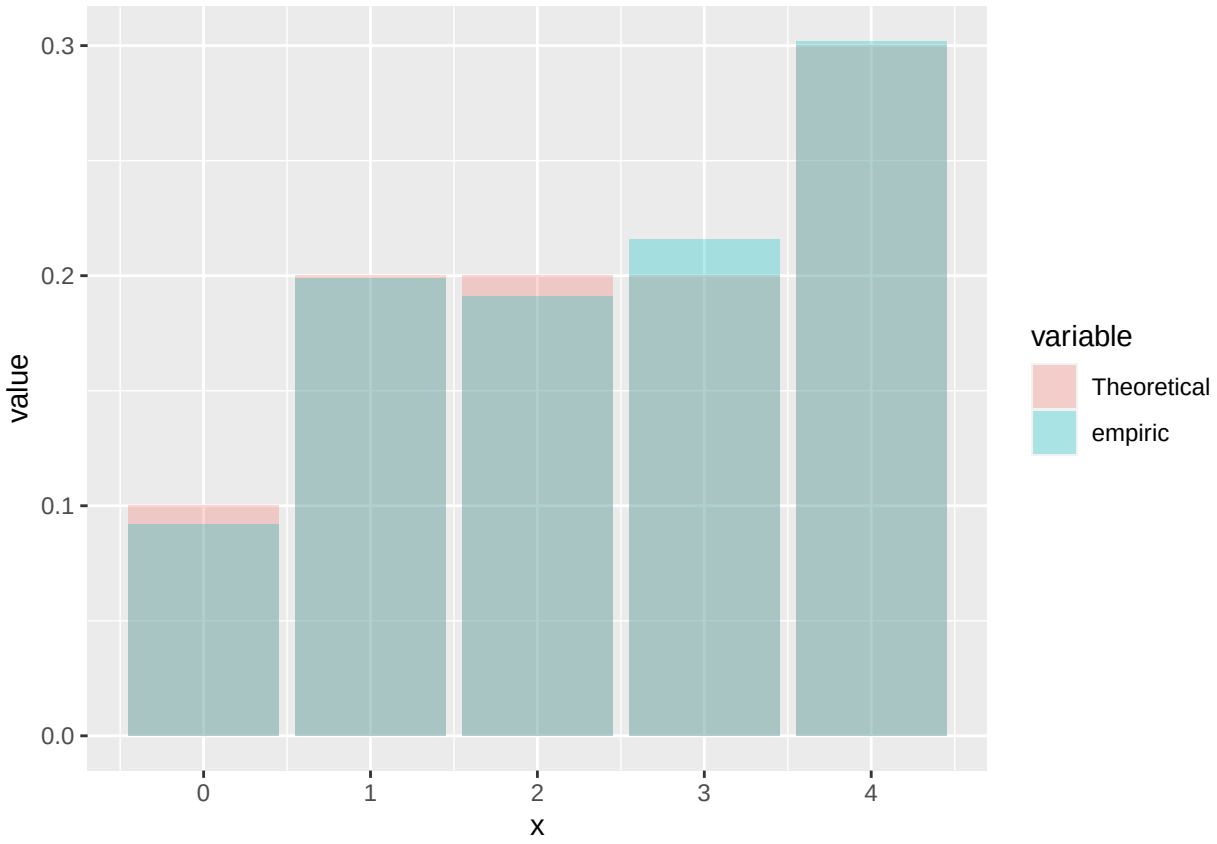
```
## Over 1000 trails of 10 spinns the probabilitie
##     of scoring a negative sum is 0.013
```

Task-3

we define a vector 'x' which we add each of the n iterations. For each iteration one random sample is drawn form U(0,1), which is by a while loop which counts the position in the vector to pull the random variable X from. eventually n RV is stored in 'x'.

the ggplot shows the comparison between the theoretical pmf given in the task and the calculated pmf trough the inverse transform.

```
set.seed(1234)
par(mfrow = c(1, 1))
X <- seq(0,4,1);
P <- c(0.1, 0.3, 0.5, 0.7, 1); #cdf of the pmf
p <- c(0.1, 0.2, 0.2, 0.2, 0.3)
n=1000 #generate 1000 samples
x <- numeric(n)

for (i in 1:n){
  counter = 1;
  r <- runif(1);
      while(r > P[counter]) #matching random uniform sample sample to cdf
          counter <- counter + 1;
          end
      x[i] <- X[counter]
  }

to_plot <- data.frame(x=X,Theoretical=p,empiric=unname(as.vector(table(x)))/n)
melted<-melt(to_plot, id="x")

print(ggplot(melted,aes(x=x,y=value,fill=variable)) +
        geom_bar(stat="identity",position = "identity", alpha=.3))
```


0.3
0.2
0.1
0.0
value
0
1
2
3
4
x
variable
Theoretical
empiric

Task-4

we lett:

$$f(x) \sim beta(3,2)$$
$$g(x) \sim U(0,1)$$

such that

$$f(x) = \frac{x^{3-1}(1-x)^{2-1}}{\frac{(3-1)!\cdot(2-1)!}{((3+2)-1)!}} = \frac{x^2(1-x)}{\frac{1}{12}} = 12 \cdot x^2(1-x)$$

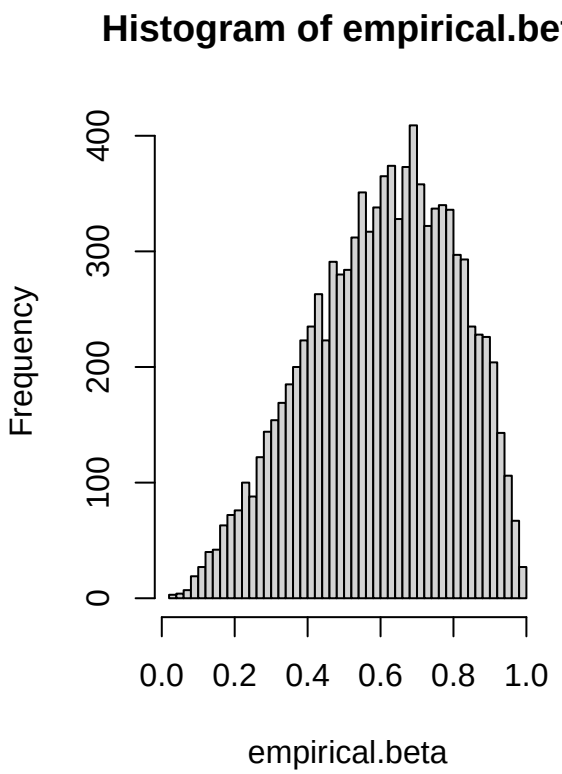
we then apply:

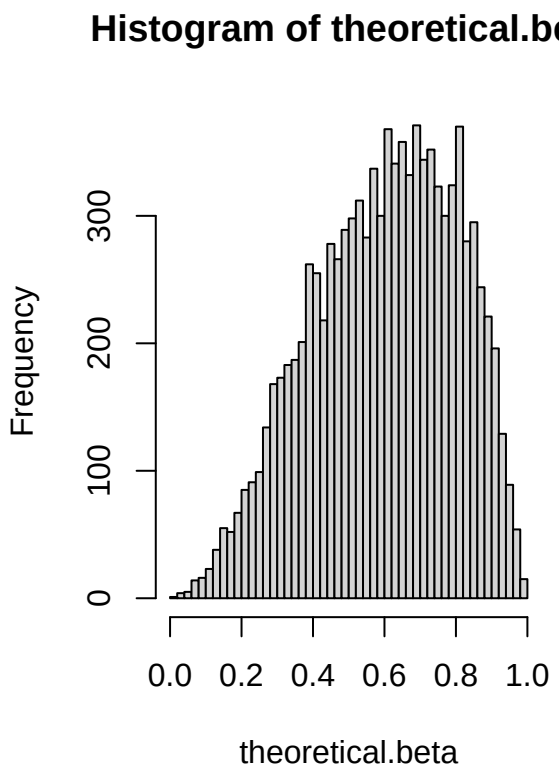
$$\frac{f(x)}{c \cdot g(x)} > u$$

and set $c = 12$ such that

$$\frac{12 \cdot x^2(1-x)}{12 \cdot U(0,1)} = \frac{x^2(1-x)}{U(0,1)} > u$$

```
set.seed(123456)
par(mfrow = c(1, 2))
n <- 10000;
k <- 0; #count of accepter
j <- 0;
y <- numeric(n)

while (k < n) {
    u <- runif(1)
    j <- j + 1
    x <- runif(1) #random variate from g
    if ((12 * x * (1-x)/12*x) > u) {
        k <- k + 1
        y[k] <- x
    }
}
empirical.beta <- y
hist(empirical.beta,freq=TRUE,breaks=50)
theoretical.beta=rbeta(n,3,2)
hist(theoretical.beta,freq=TRUE, breaks=50)
```


Histogram of empirical.beta
Frequency
0
100
200
300
400
0.0
0.2
0.4
0.6
0.8
1.0
empirical.beta


Histogram of theoretical.beta
Frequency
0
100
200
300
0.0
0.2
0.4
0.6
0.8
1.0
theoretical.beta

Task-5

$$\frac{f(x)}{c \cdot g(x)} = \frac{\frac{1}{\sqrt{(2pi)}} exp\{-\frac{x^2}{2}\}}{c \cdot \frac{1}{2} exp\{-|x|\}} = \frac{2}{c \cdot \sqrt{(2pi)}} e^{\{|x| - \frac{x^2}{2}\}}$$

$$\frac{f(x)}{c \cdot g(x)} < 1$$

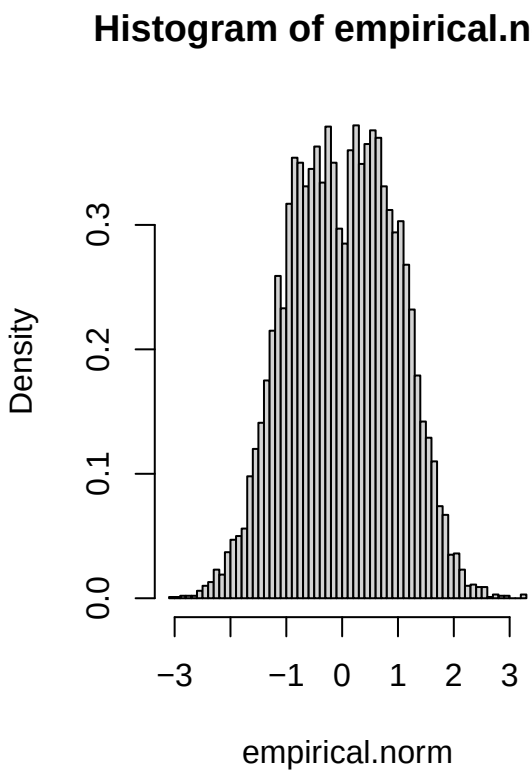
c is set to 2, to enshure that:

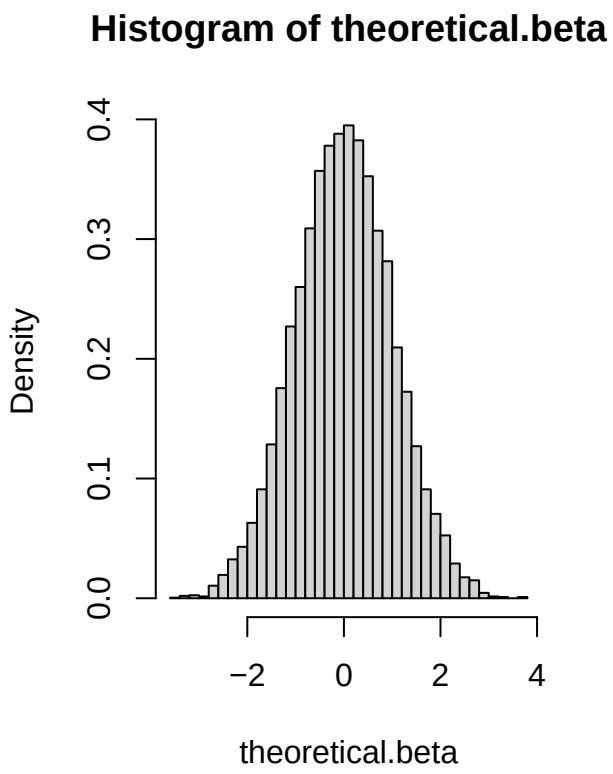
$$\frac{f(x)}{c \cdot g(x)} < 1$$

since $f(x) \sim N(0,1)$ we sample the input x from normal distribution and

the accept criteria v from uniform.

```
set.seed(123456)
par(mfrow = c(1, 2))
n <- 10000;
k <- 0; #count of accepter
j <- 0;
y <- numeric(n)

while (k < n) {
    v <- runif(1)
    j <- j + 1
    x <- rnorm(1)
    if ((1/sqrt(2*pi))*exp(abs(x)-((x^2)/2)) > v) {
        k <- k + 1
        y[k] <- x
    }
}
empirical.norm <- y
hist(empirical.norm,freq=FALSE, breaks=50)
theoretical.beta=rnorm(n)
hist(theoretical.beta,freq=FALSE, breaks=50)
```


Histogram of empirical.norm
Density
0.0
0.1
0.2
0.3
−3
−1
0
1
2
3
empirical.norm


Histogram of theoretical.beta
Density
0.0
0.1
0.2
0.3
0.4
−2
0
2
4
theoretical.beta

Task-6

psaudo kode: first sample from uniform distribution

$$U = \{U_1, U_2, ..., U_n\} \sim U(0,1)$$

apply which(as indicator function)

$$m = n(U \cdot 1_{\{p>u\}})$$

and use the cardinality trough length() function of the vector sorted out the two subsets to sample the two normal distributions acc their weighting p

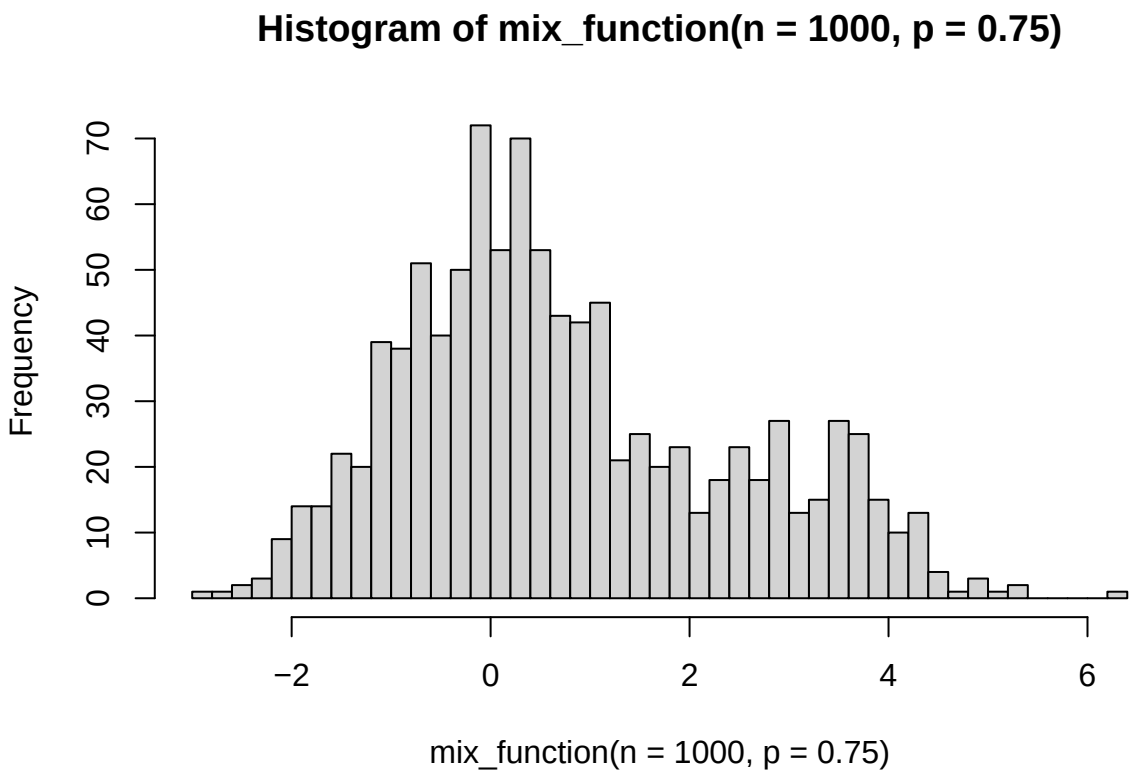
$$N_{N(0,1)} = \{N_1, N_2, ..., N_m\}$$
$$N_{N(3,1)} = \{N_1, N_2, ..., N_{n-m}\}$$

yout then have the right prortion of the mixture

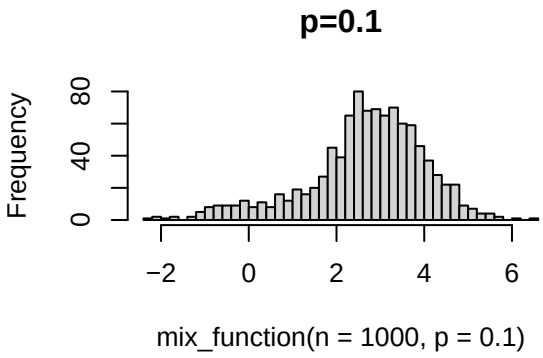
$$p \cdot N(0,1) + (1-p) \cdot N(3,1)$$

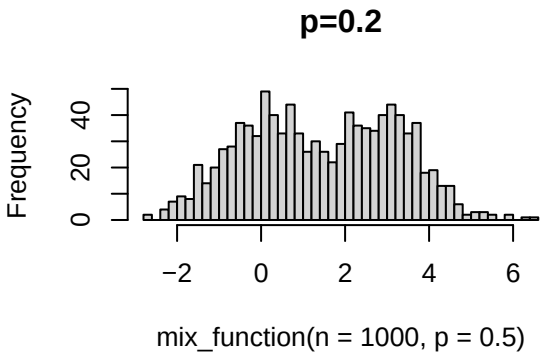
result of the histograms below show that an approximate 50:50 mixture of the two distributions will give a bimodal distribution.

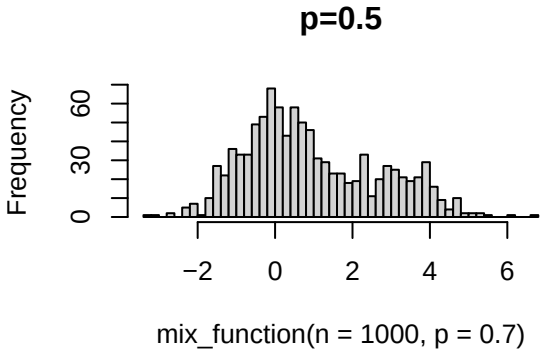
```
set.seed(123)


mix_function <- function(n, p){
    U <- runif(n)
    m <- length(which(p > U))

    N1 <- rnorm(m,0,1)
    N2 <- rnorm(n-m, 3, 1)

    Z <- c(N1, N2)
}

hist(mix_function(n=1000, p=0.75), breaks = 40)
```

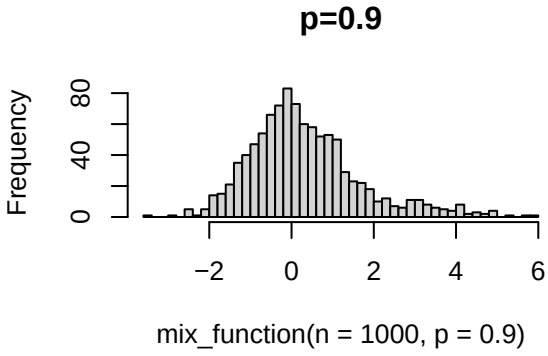



```
par(mfrow=c(2,2))
hist(mix_function(n=1000, p=0.1), breaks=40,main='p=0.1')
hist(mix_function(n=1000, p=0.5), breaks=40,main='p=0.2')
hist(mix_function(n=1000, p=0.7), breaks=40,main='p=0.5')
hist(mix_function(n=1000, p=0.9), breaks=40,main='p=0.9')
```


p=0.1
Frequency
0
40
80
−2
0
2
4
6
mix_function(n = 1000, p = 0.1)


p=0.2
Frequency
0
20
40
−2
0
2
4
6
mix_function(n = 1000, p = 0.5)


p=0.5
Frequency
0
30
60
−2
0
2
4
6
mix_function(n = 1000, p = 0.7)


p=0.9
Frequency
0
40
80
−2
0
2
4
6
mix_function(n = 1000, p = 0.9)

Task-7

The idea is to sample $N(t) \sim pois(\lambda(t))$ for $t = 10$,
and $N(t)$ times $Y \sim gamma(\alpha, \beta)$ which we sum over n amounth of times. And then re run this function fro several λ. Use the first gamma of each iteration to produce the theoretical values.

It ended up in the less charming chunk below, with the same loop twice with different λ and the same (α, β) however the result seem plaucible enough.

"find som popcorn and wait til the code has runned - Enjoy!"

```
t <- 10;
lmbd1 <- 1;
lmbd2 <- 2;

alpha <- 5
beta <- 4
n <- 1000
t <- 10

          X1 = numeric(n)
          E.Y1 <- numeric(n)
          E.Y1.sq <- numeric(n)



          for(i in 1:n){
                  n.t<- rpois(n=1, lmbd1*t)
                  Y <- rgamma(n=n.t, alpha, beta)
                  X1[i] <- sum(Y)
                  E.Y1[i] <- mean(Y)
                  E.Y1.sq[i] <- mean(Y^2)
          }

          th_mean1 <- t*lmbd1*E.Y1[1]
          th_variance1 <-t*lmbd1*  E.Y1.sq[1]


          X2 = numeric(n)
          E.Y2 <- numeric(n)
          E.Y2.sq <- numeric(n)



          for(i in 1:n){
                  n.t<- rpois(n=1, lmbd2*t)
                  Y <- rgamma(n=n.t, alpha, beta)
                  X2[i] <- sum(Y)
                  E.Y2[i] <- mean(Y)
                  E.Y2.sq[i] <- mean(Y^2)
          }

          th_mean2 <- t*lmbd2*E.Y2[1]
          th_variance2 <-t*lmbd2*  E.Y2.sq[1]
```

```
cat('mean and variance is', mean(X1),'and',var(X1),
    'theoretical mean\n and variance', th_mean1, 'and', th_variance1,
    '\n with lambda=',lmbd1,'\n','\n',
    'mean and variance is', mean(X2),'and',var(X2),
    'theoretical mean\n and variance', th_mean2, 'and', th_variance2,
    '\n with lambda=',lmbd2)
```

```
## mean and variance is 12.53467 and 16.73411 theoretical mean
##  and variance 10.55847 and 12.7002
##  with lambda= 1
##
##  mean and variance is 24.96344 and 38.79678 theoretical mean
##  and variance 26.27482 and 38.43432
##  with lambda= 2
```

Task-8

Grunnet noe tidspress på slutten (kryssende oblig) ble denne stående åpen.

Task-9

```
CI <- function(n, my, alpha){
  X <- rnorm(n, my)

  X.bar <- (1/n)*sum(X)         #sample average
  S <- sd(X)                    #Sample std. deviation

  error <- qt(p = 1-alpha/2, n-1) * (S/sqrt(n)) #quantile error

  CI.upper <- X.bar + error
  CI.lower <- X.bar - error

  return(c(CI.lower, CI.upper))
}

cat('CI(Upper, Lower)=', '(', CI( n=100, my = 0, alpha = 0.05 )[1],
                              CI( n=100, my = 0, alpha = 0.05 )[2],')')
```

```
## CI(Upper, Lower)= ( -0.03166469 0.2032248 )
```

Task-10

we are to test if: $H_0 = 100$ or

$H_1 \neq 100$

for a significance level $\alpha = 0.05$

for a sample size $N = \{10, 20, 30, 40, 50\}$

and verifying the quality of the test by reampling it and plot the power curve.

The Power denoted $(1 - \beta)$ is the probabilitie of rejecting H_0 given

that H_1 is correct.

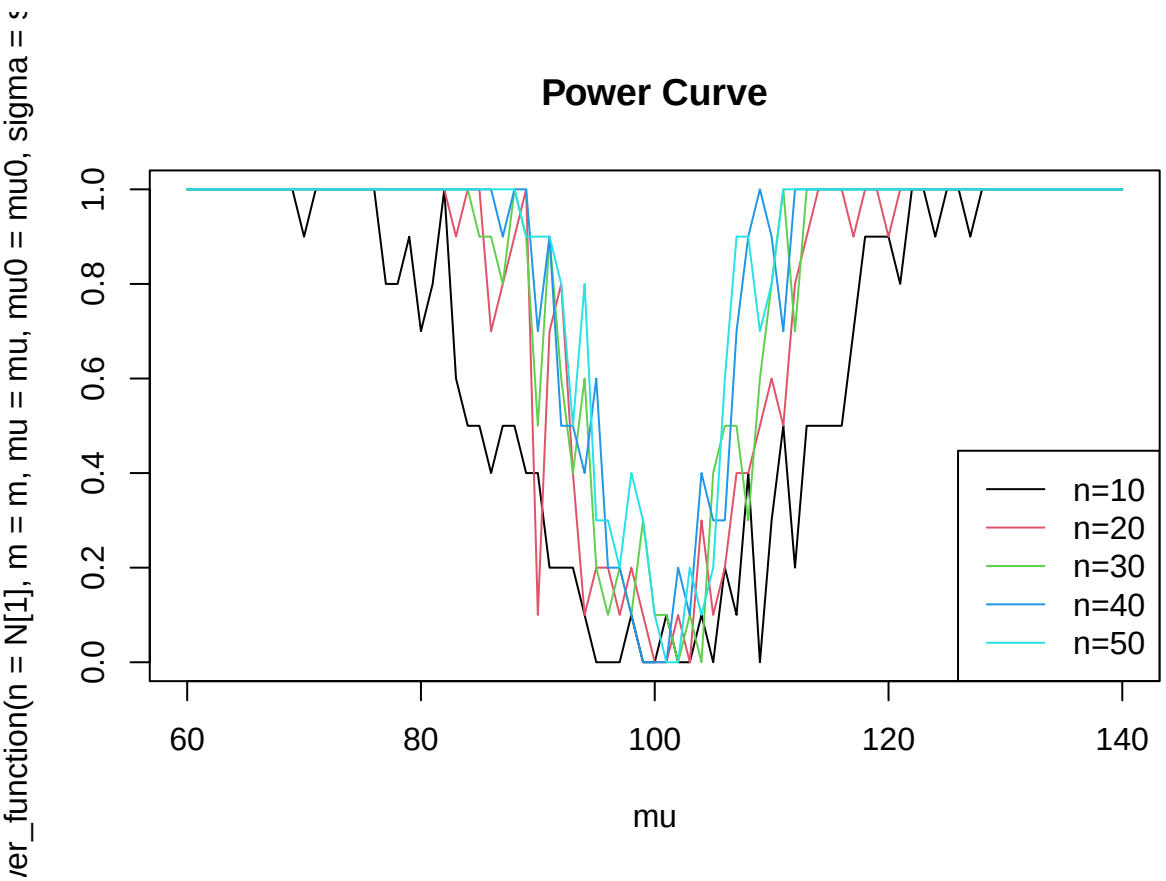
To simulate this we will resample the test under several $\mu = (60, 61, ..., 140)$

and redo this for several sample sizes N.

As we see there is a much more noisie picture for smaler sample sizes.

hence it is a waker test acc to the power test.

```
N <- c(10,20,30,40,50)
m <- 10
mu0 <- 100
sigma <- 20
mu <- c(seq(60,140,1))

power_function<-function(n, m, mu0, mu, sigma){

    M <- length(mu)
    power <- numeric(M)
    for (i in 1:M) {
      mu1 <- mu[i]
      pvalues <- replicate(m, expr = {
            x <- rnorm(n, mean = mu1, sd = sigma) #sample n sized set of data to test
        ttest <- t.test(x, alternative = "two.sided", mu = mu0)
        ttest$p.value  } )                  # Replicate test m times
      power[i] <- mean(pvalues <= .05)      # Avrage the p values that are positive

    }
      return(power)
}


plot(mu,power_function(n=N[1], m=m, mu=mu, mu0=mu0, sigma=sigma),type='l', main = 'Power Curve',)  ##th
lines(mu,power_function(n=N[2], m=m, mu=mu, mu0=mu0, sigma=sigma),col=2)
lines(mu,power_function(n=N[3], m=m, mu=mu, mu0=mu0, sigma=sigma),col=3)
lines(mu,power_function(n=N[4], m=m, mu=mu, mu0=mu0, sigma=sigma),col=4)
lines(mu,power_function(n=N[5], m=m, mu=mu, mu0=mu0, sigma=sigma),col=5)
legend("bottomright", legend = c('n=10','n=20','n=30','n=40','n=50'),lty=1,col=1:5)
```


er_function(n = N[1], m = m, mu = mu, mu0 = mu0, sigma = s
Power Curve
1.0
0.8
0.6
0.4
0.2
0.0
60
80
100
120
140
mu
n=10
n=20
n=30
n=40
n=50